POLITECNICO
MILANO 1863

Requirements Analysis and Specifications Document

version 1.1

13rd November 2016

Authors:
Davide Moneta (808686)
Federico Oldani (806337)
Luca Oppedisano (878301)

Index

1 Introduction

1.1 Purpose

In our Requirement Analysis and Specification Document we are going to describe the system we modeled for the PowerEnJoy car sharing service.

We will describe all the features we want to implement, we will deal with the functional and nonfunctional requirements needed to have a working system and with the conditions in which our software can operate, we will present what we assume to be the necessities of our users and most of the supported use cases, and there will be simulations of some situations that could occur when using our services.

This document is addressed to developers and programmers who will have to implement the software, to system analysts who are willing to integrate other system with ours, and can be used as a contractual base between the employer and the developers.

1.2 Scope

We will design a software to manage the functionalities of PowerEnJoy: a new car sharing service. PowerEnJoy offers exclusively electric cars that allow the users to move around respecting the environment.

Using this platform customers are able to find and reserve one of our cars within a certain distance from their current location or a specified address. When they reach the chosen car, they can unlock it and drive it.

Users are charged with a per-minute fee, but the company incentivizes virtuous behaviors by offering a discount at the end of a trip if certain conditions are met.

1.3 Actors

Registered users
They can log in the platform, see the available cars around a specified location, reserve a car, unlock a car they have reserved, drive the car, navigate to a location, pause the trip (if they need the car again after a stopover), access the map of charging areas, park the car, plug the car into the power grid at any SPA
Service personnel

They will take care of re-charging cars distant from the SPAs and they will answer to users' help request

1.4 Glossary: definitions, acronyms, abbreviations

Visitor: a generic person who access the website or the app without being authenticated
User: equivalent to registered user, a visitor who already performed the registration and has now access to the system
Car: with this term we always refer to one of our cars
Car tag: a term to label the status of a car, it can be "available" or "reserved" or "in use"
Available car: a car that is currently not in use nor reserved by anyone and has at least 30% of battery level
Reservation: it can be requested by a user for an available car, this gives the user the possibility to unlock the reserved car within a time span of 1 hour from the reservation time, after this time the reservation expires
Reserved car: a car that can be unlocked only by the one who reserved it
Out of power: this is the tag of the cars with less than 30% battery level and nobody is using it, this cars appear only in the assistance map.
Ignition code: this code is sent to the user when a reservation is successful, user will have to type this code on the OBS lock screen in order to ignite the engine
Unlocking request: an action that can be performed by someone who reserved a car, this action grants access to the car; users can do this by tapping the dedicated button in the app, if they are close enough the the vehicle they want to unlock and if they reserved it. If the unlock is successful the car is tagged as "in use"
Locking a car: after the passengers are all out of the car and the door are all closed the system counts 40 seconds and then locks the car. There are two types of locking: if the user wants to keep the car on pause, the car is locked and it remains on pause for that user, otherwise the car is locked and made available to other users. System locks the car only if the car is inside the Safe Area
Pause: when the user turns off the engine the systems asks him on the OBS if he wants to put the car on pause, if the user taps "YES" the pause state is applied: the user keeps paying the per-minute fee until he unlocks it again
Ride or Trip: it starts when the engine is ignited or 2 minutes after the car is unlocked, and ends when the car is locked
Temporary bill: keeps track of the duration of the trip or the pause to calculate the amount of money to pay

The Area: it's the area equivalent to the city where PowerEnJoy is active and where you can find our cars, you can drive out of The Area but if you want to park out of it you can only put the car on pause
Safe area: we consider a safe area the set of every possible legal parking spot within The Area, so all the places where it is possible to park the car without being fined
SPA: these are Special Parking Areas where cars can be parked and also recharged through a special plug
Virtuous behaviour: we consider such behavior those that prevent extra cost for the company and we offer some discounts to users who perform these
The System: the personification of the software, which manages all the functionalities of PowerEnJoy
MSM: Money Saving Mode, this can be enabled in the car after selecting a destination, this ensures to find parking solution to get a discount, optimizing the cost of the trip; the solution takes also into account the position of cars in The Area to ensure a uniform distribution
OBS: On Board Screen, it's the display inside the car, where the user can visualize informations about the car and the trip

1.5 Reference documents

Assignment document: AA 2016-2017 Software Engineering 2 - Project goal, schedule, and rules "PowerEnJoy"
ISO/IEC/IEEE International Standard 29148-2011- Requirements engineering
RASD samples:

- A.A. 2014-2015 Software Engineering 2: "MeteoCal"
- A.A. 2015-2016 Software Engineering 2: "myTaxiService"
- A.A. 2012-2013 Software Engineering 2: "SWIM v2"

1.6 Overview

The remainder of the document is structured like follows:

2 - Overall Description: here we give a high level description of the software and we present the offered functionalities; we show in which constraints the system can operate and what assumptions we did on the world
3 - Specific Requirements: chapter dedicated to all kind of interfaces involved in the system and to functional and nonfunctional requirements
4 - Scenario Identifying: a collection of possible situation that may occur when the service will be operational

5 - UML Models: the section where we present most of our models, like use cases, class diagrams, sequence diagrams, state chart
6 - Alloy modelling: where we prove the consistency of our model of the world
7 - Used Tools: contains the list of the tool we used to complete the RASD
8 - Hours of work: where we kept track of the time spent on the RASD

2 Overall Description

2.1 Product perspective

The system we are modeling is a standalone software, which is able to interface with PowerEnJoy cars to provide all the features of the car sharing service. It's developed as a web application, accessible through a browser and easily convertible into a native mobile application. Currently there is no intention to make any API available for third-party integration.

2.2 Product functions

All the functions of our system can be expressed through the following goals:

- ➔ [G1] Allow visitors having a valid driving license and a valid payment option to register to the system and so to become a user
- ➔ [G2] Allow users to log into the system
- ➔ [G3] Allow users to find a car within a certain distance (chosen by the user) from a specified location or from their current location
- ➔ [G4] Allow users to reserve a car up to an hour before picking it up
- ➔ [G5] If the reserved car is not picked up, charge the user 1€ and the reservation expires
- ➔ [G6] Allow to unlock a car
- ➔ [G7] Allow users to drive the reserved car and to know the current bill based just on trip time, with the discount applied only at the end of the trip
- ➔ [G8] Allow users to know where the nearest SPAs are located
- ➔ [G9] Allow users to enable MSM to reach their destination paying as little as possible
- ➔ [G10] Allow users to plug cars into SPAs' power grid and get the discount of 30% on the last ride
- ➔ [G11] Lock the car after user parks and leaves the car
- ➔ [G12] Allow users to park and pause the trip to keep the car reserved if they have to go for a stopover
- ➔ [G13] Allow users to contact the support by a phone call, if they need assistance

- ➔ [G14] Allow staff to know where the PowerEnjoy cars are, their battery level and reservation history
- ➔ [G15] Allow staff login to the system
- ➔ [G16] If the system detects at least two passengers (driver excluded), the system applies a discount of 10% on the last ride
- ➔ [G17] If the car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride
- ➔ [G18] If car is left at more than 3 km from the nearest power grid station or with more than 80% the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site

2.3 User characteristics

The target users of this service are those people who believe in the advantages of car sharing: a convenient service when you don't own a car and you need one just for a brief period, as an alternative to public transport; in our specific case it's a good choice also for those who can't afford an electric car but still want to respect the environment.

2.4 Constraints

The software will operate within these constraints:

2.4.1 Regulations

The system must ask the user permission to get the position of the car in which they are and to manage their sensible data. The system will send to the user only the kind of notifications that he agreed to receive. The system must ask the user permission to access their credit card in order to accomplish payments.

2.4.2 Hardware limitations

Minimum requirements for the developers depend on the tools used to build the web application.

2.4.3 Interfaces to other applications

The software will interface with the car system, to read data from sensors like position, battery charge, weight on the seats.
There will be an interface with a database through a DBMS, to retrieve and store data about the cars and the SPAs.
The system will interface with the PayPal payment service to accept users payments.

Finally, the system will interface with the mobile operative systems' APIs to allow push notification.

2.4.4 Parallel operations

More than one user will be using the platform at the same, so it is necessary to manage their operations concurrently at an application and database level.

2.4.5 Control functions

Our software will control the locking and unlocking process with regard to the cars, and the billing procedure.

2.5 Assumptions and Dependencies

Our requirements and goals are based on the following assumptions on the world, with regard to what can and cannot happen, and on the specifications, with regard to the solutions we found to solve unspecified cases:

Domain properties

- Cars have a positioning system that always works
- Cars can always communicate with the system through a dedicated 3G connection
- The System is always able to get cars' location with a 5 meters accuracy
- Users never lies about:
    - Their payment informations
    - Their personal data
    - Their driving license number and validity
- Users' bandwidth is always good enough to access the application
- There's always an operator available to answer to assistance calls
- Weight sensors in the car can always determine correctly the number of passengers on board
- There is always at least one car available to pick up
- The person who reserved the car is responsible for what happens during the trip

Text assumptions

- When the user leaves the car in a SPA, before completing the payment, we give him 2 minutes to plug the car into the power grid, so he will be able to get a discount
- The car keeps track of the current amount of money to pay in a local bill, at the end of the trip a discount may be applied, then the bill is sent to the system and the payment is carried out

- All cars are the same model
- User will have 2 minutes to ignite the engine after they unlock a car, after this time they will start being charged
- If the user is eligible for more discounts we apply only the higher one
- Battery charge percentage is referred to the total capacity of the battery
- The SPAs are the only way to charge a car
- Car will always have at least 30% of battery charge, we pay people to recharge them if they are under that threshold because we don't want our user to pay more if it's not their fault

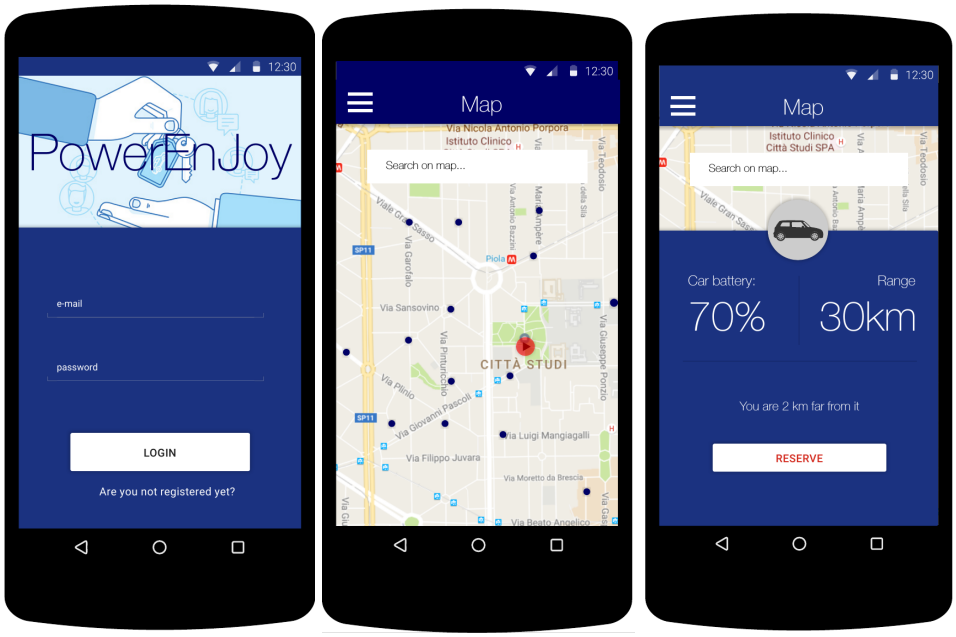
3 Specific Requirements

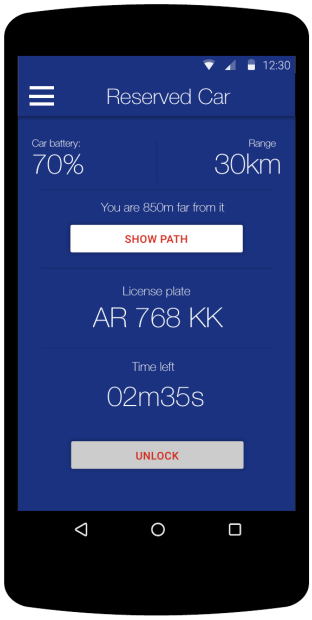
3.1 External Interface Requirements

3.1.1 User Interfaces

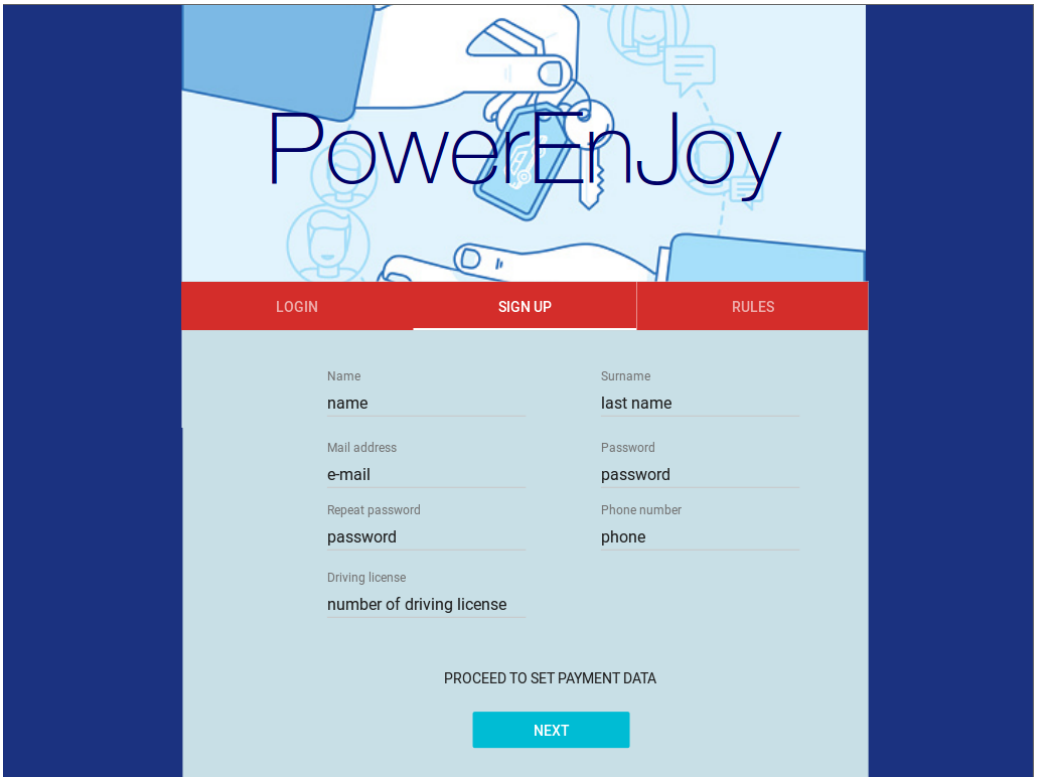
Here are presented some mockup of user interface:

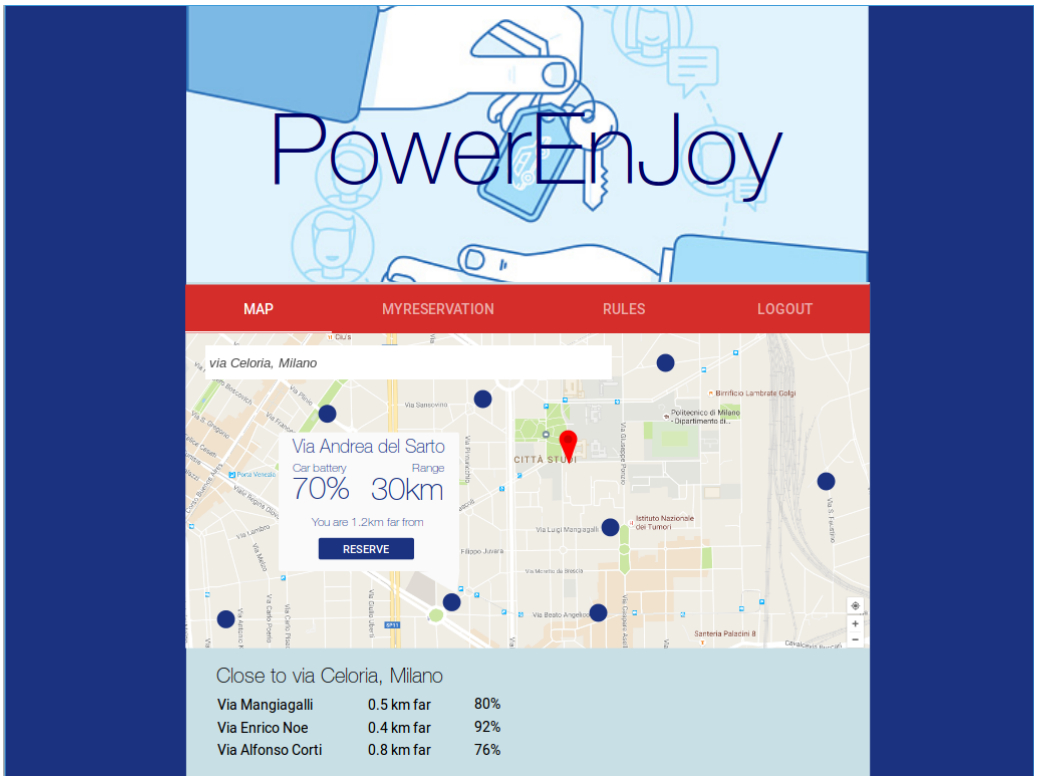
3.1.1.1 Mobile application





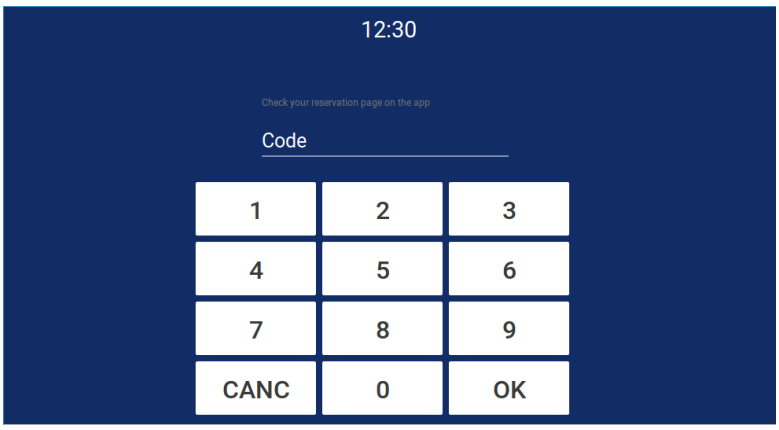
3.1.1.2 Web application

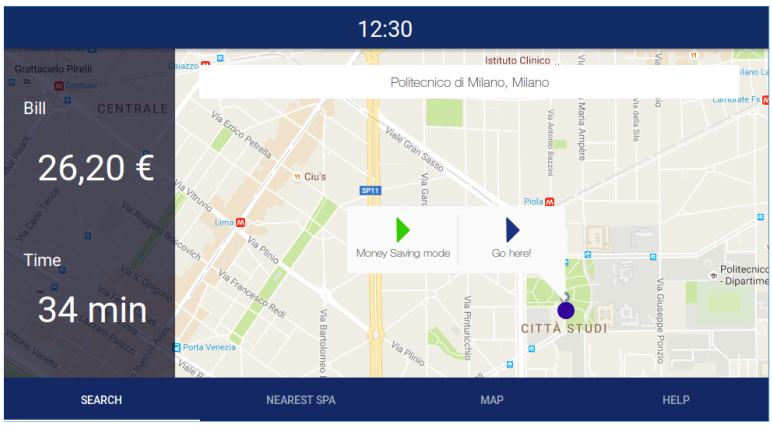


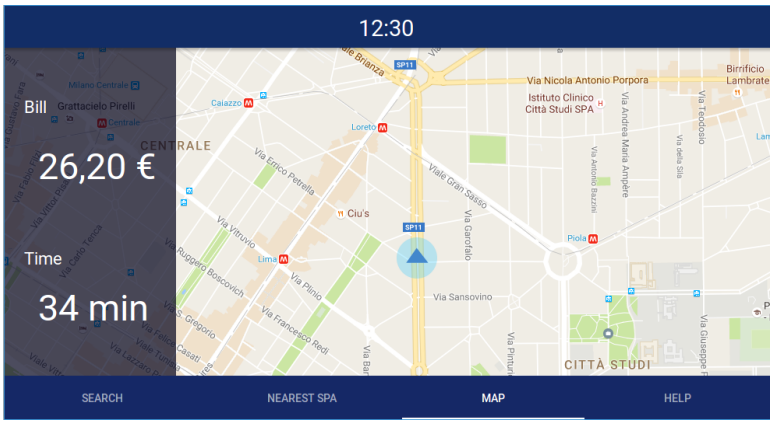


3.1.1.3 On Board Screen (OBS):

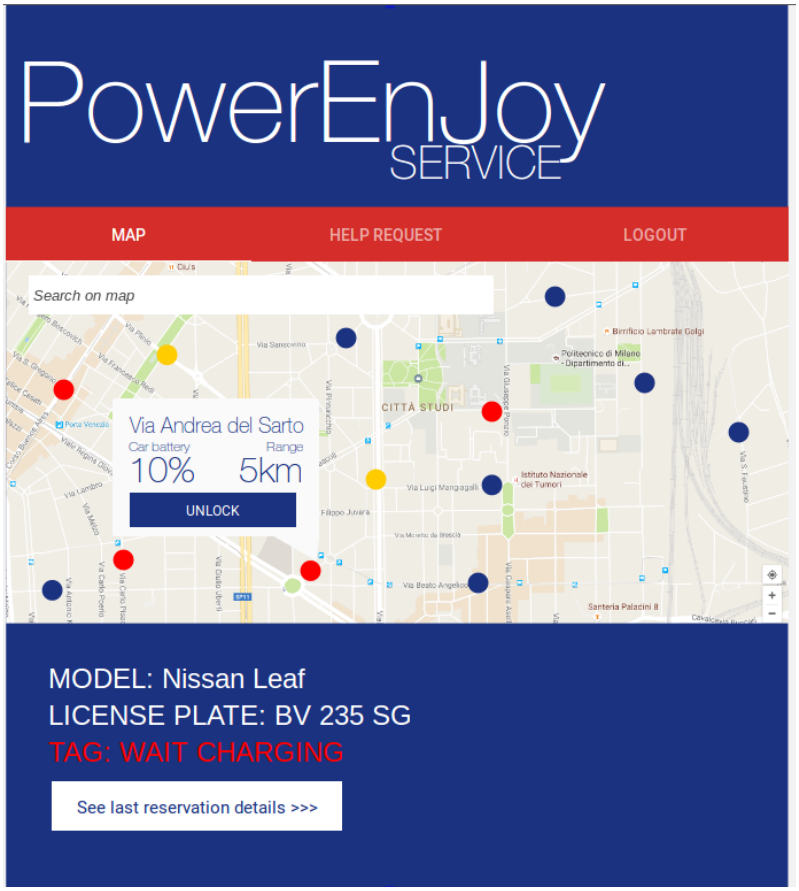
The user must insert a code, given by TS through the app, into the lock screen on the "On Board Screen" in order to let the engine start. He can look for a destination and decide to go there or activate MSM. Moreover system on board provides navigation to the nearest SPA and a Help button to call assistance.




12:30
Politecnico di Milano, Milano
Bill
26,20 €
Time
34 min
Money Saving mode
Go here!
CITTÀ STUDI
SEARCH
NEAREST SPA
MAP
HELP


12:30
Bill
26,20 €
Time
34 min
CITTÀ STUDI
SEARCH
NEAREST SPA
MAP
HELP

3.1.1.4 Support operator interface



3.1.2 API interfaces

3.1.2.1 Map

We need a map with multiple markers for cars and SPAs and a navigation system. Due to the cost, we decided not to develop a navigation system in house but to use APIs provided by HERE maps. For more information see https://developer.here.com

3.1.2.1.1 Safe and Unsafe area

We will need a third party software library which can provide informations about certain coordinates. It will tell us if the specified position is in The Area and if it's in a Safe or Unsafe Area.

3.1.2.2 Payment system

We will take advantage of the PayPal APIs to allow our clients to pay their bills. PayPal APIs offer the possibility to pay through user's PayPal account or credit card. For more information refer to https://developer.paypal.com/

3.1.3 Architectural considerations

We will use an SQL database to store all data, HTML, CSS and javascript for the web application, JAVA to develop mobile application. Furthermore we will use an application server to manage the users requests.

3.2 Functional Requirements

[G1] Allow visitors having a valid driving license and a valid payment option to register to the system and so to become a user

[R1] Visitors must not be already user.
[R2] Visitors must insert email address, password, driving license number, payment data and phone number
[R3] Visitors can see login, registration and information pages only
[R4] Visitors must receive an email at the address provided during the registration phase with a link to activate their account
[D1] Email address and driving license used for registration must be formally correct
[D2] Payment data must be correct

[G2] Allow users to log into the system:

[R1] User must insert email address and password which have been provided in the registration phase
[R2] Wrong password, wrong email address or a wrong combination of this two must not grant access to the service

[G3] Allow users to find a car within certain distance (chosen by the user) from a specified location or from their current location

[R1] The address entered by the user must be highlighted on the map
[R2] All available cars around the chosen address must be shown
[R3] User position must be shown on the map
[R4] The system must know each available car position through car's GPS
[R5] Battery level of each car must be displayed in the map

[G4] Allow users to reserve a car up to an hour before picking it up

[R1] User must not have pending payments with the company
[R2] The user must not have other active reservations
[R3] The user isn't using other PowerEnJoy cars
[R4] After reservation, the time remaining before expiration must be visible both in the webapp and in the mobile app
[R5] After reservation, the user must be able to see the ignition code necessary to ignite the engine

[G5] If the reserved car is not picked up, The System charges the user 1€ and the reservation expires

[R1] The car has not been unlocked during the reservation period of 1 hour
[R2] The system must charge the user 1€
[R3] The system must set the car as an available car

[G6] Allow to unlock a car

[R1] The user has to send an unlock request from the mobile application
[R2] The system must know cars positions
[R3] The system must detect user device's position
[R4] The user position must be within 10 meters from car position
[R5] The unlock request can be sent only by whom reserved or put on pause that car
[R6] If the user unlock the car that he put on pause, System charge him for the pause time

[G7] Allow users to drive the reserved car and to know the current bill based just on trip time, with the discount applied only at the end of the trip

[R1] The system must show the temporary bill on the OBS
[R2] The user has entered the temporary code (provided by the app at the moment of reservation) on the lock screen
[R3] The system must know the trip duration
[D1] Each car has weight sensors to detect how many people there are

[G8] Allow users to know where the nearest SPAs are located via OBS

[R1] The system can retrieve car position with a 5 meters accuracy
[R2] The system must provide a button on the OBS to start the research of the nearest SPAs
[R3] The system must show SPA details on the OBS

[R4] The on board system must know where SPAs are located

[G9] Allow users to enable MSM to reach their destination paying as little as possible

[R1] The system must know parked car position with a 5 meters accuracy
[R2] User has selected the MSM on the on board screen
[R3] The on board system must know where SPAs are located
[R4] The system must know cars distribution within The Area and will send to the on board system the coordinates of the selected SPA

[G10] Allow users to plug cars into SPAs' power grid and get the discount of 30% on the last ride

[R1] The car must be parked in a SPA
[R2] The system must know where SPAs are located
[R3] The charging cable is plugged in correctly
[R4] The discount is applied if it's the only one or if it's the higher one
[R5] System must know the correct cars positions
[R6] The doors of the car must be closed
[R7] Weight sensors must not detect people in the car

[G11] Lock the car after user parks and leaves the car

[R1] Weight sensors must not detect people in the car for 40 seconds, if car is in SPA timer is set to 120 seconds to allow user to plug charging cable.
[R2] The doors of the car must be closed for 40 seconds, if car is in SPA timer is set to 120 seconds to allow user to plug charging cable.
[R3] The system must charge the user
[R4] System lock the car
[R5] If user didn't select pause option and the battery level is more than 30%, the car is set as available
[R6] If the battery level is less than 30%, car must be tag as out of power
[R7] If the car is in The Area, it must be in Safe Area
[D1] Each car has weight sensors to detect how many passengers are inside
[D2] Each car has sensors to detect open doors

[G12] Allow users to park and pause the trip to keep the car reserved if they have to go for a stopover

[R1] The user can tap pause option on the OBS when engine was turned off
[R2] The user must be charged for the time travelled before the pause in order to confirm the pause operation
[R3] The car has to be out of a SPA
[R4] Weight sensors must not detect people in the car
[R5] The doors of the car must be closed
[D1] Each car has weight sensors to detect how many people are inside
[D2] Each car has sensors to detect open doors

[G13] Allow users to contact the support by a phone call, if they need assistance

[D1] Each car is always in the conditions to start a phone call
[R1] OBS must have a button dedicated to assistance call
[R2] The app must have a button dedicated to assistance call
[R3] Assistance can know where cars are

[G14] Allow staff to know where the PowerEnjoy cars are, their battery level and reservation history

[R1] System must know the position of each car
[R2] System must know battery level of each car
[R3] System must store all the reservation requests and usage data for each car
[R4] Staff can see all car details selecting a car from map

[G15] Allow staff login to the system

[D1] Staff credentials are entered in the database by the company
[R1] The combination of email address and password typed is correct
[R2] Staff logged in can answer help request, see cars positions and unlock the cars

[G16] If the system detects at least two passengers (driver excluded), the system applies a discount of 10% on the last ride

[R1] The weight sensors must detect at least 3 people in the car
[R2] The car must be locked
[R3] The discount is applied if it's the only one or if it's the higher one

[G17] If the car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride

[D1] On board system knows car battery level
[R1] Battery level must be less than 50%
[R2] The car must be locked
[R3] The discount is applied if it's the only one or if it's the higher one

[G18] If car is left at more than 3 km from the nearest power grid station or with more than 80% the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site

[D1] On board system knows car battery level
[R1] System must know the position of the car
[R2] System must know SPAs locations
[R3] Battery level is less than 20% OR the nearest SPA is 3 km far
[R4] The car must be locked

3.3 Non-functional Requirements

3.3.1 Performance Requirements

Performances of The System must be suitable to guarantee an acceptable grade of usability. The waiting time for the mobile app and for the web page response are assumed null (or eventually tending to zero). Thence the performance issues are enclosed by the connection speed of the user's internet provider.

3.3.2 Design Constraints

3.3.2.1 Hardware limitations

3.3.2.1.1 Smartphone app

The app needs these minimum requirements to run without any problem:

- GPS/GLONASS connection
- 3G or 4G connection
- enough space on phone's memory to safely install the app

3.3.2.1.2 Web app

The website needs an internet connection.

3.3.2.2 Software System Attributes

3.3.2.2.1 User friendliness

Both the web app and the smartphone app are easy to use allowing even novices to use The System.

3.3.2.2.2 Reliability

Every part of The System (Servers, cars, users' smartphones and users' PCs using the web app) is connected to the internet so every change at every significant part is immediately shared within The System and pushed to all the other parts.
The car position accuracy is guaranteed by a GPS sensor onboard that can give the exact position of the vehicle with an error of ±10m everywhere on Earth.
The system that recognize the current battery charge is embedded in the electric car so we assume its reliability.
The previous assumptions guarantee a significant level of reliability in every aspect of The System's operation.

3.3.2.2.3 Availability

The System will be available 24/24h and 7/7d. To do so there is a system that permits a high level of redundancy.

3.3.2.2.4 Security

The System is able to protect security of the users data.

3.3.2.2.5 Maintainability

The code is exhaustively described and commented in order to be easily modified in future.

3.3.2.2.6 Portability

The System portability is guaranteed in these environments:
Smartphone app
User smartphone's OS needs to be Android (version 5.0 or above), iOS (version 9.0 or above) or Windows Phone (version 8.1 or above).
Web app
User PC needs to have a browser that permits the use of AJAX.

4 Scenario identifying

Here are some examples of use of TS:

4.1 Scenario #1 (Sign up)

Pippo is a person with a regular valid driver license who wants to use TS to rent a car for his travels around the city. After connecting to Web App he clicks over the button “Sign up” and compiles the form with his name, surname, date of birth, residence address, email address, telephone number, driver licence number and enables PayPal payment to TS.
TS adds Pippo’s data to the database and generates pseudorandomly a password that is sent to Pippo’s e-mail in order to make possible for him to access TS.

4.2 Scenario #2 (Log in)

Pippo, after signing up, decides to use TS to rent an electric car. To do so he opens the PowerEnJoy app on his Google Pixel smartphone and, after writing his e-mail and password, clicks on the “Log in” button.

4.3 Scenario #3 (normal use)

Rodolfo Salvi logs in, reserves a car near the Politecnico, after 10 minutes he comes close to it, unlocks it and gets into it. He ignites the engine and drives to an indian restaurant, then he parks, gets out of the car and locks it; the bill is charged on his credit card.

4.4 Scenario #4 (reservation expires)

Massimo Decimo Meridio, after logging in the web application, decides to reserve a car parked right under his house, he goes for a little nap but he forgets to set the alarm, and when he wakes up, 1 hour e 1 minute later, his reservation is expired, he has been charged 1€, the car is shown on the map as available and he has to reserve it again.

4.5 Scenario #5 (fast unlock)

John Cena is going towards the train station because he needs to go to work, but suddenly he notices a PowerEnJoy car parked in front of him, he opens the app to check if it's already reserved, and with surprise he finds out that it's available, so with two taps he reserves and unlocks it; then he drives to the arena and parks the car in a regular parking. The payment is carried out successfully.

4.6 Scenario #6 (payment failed)

Danilo TheBank has just finished his PowerEnJoy trip and, after having parked in a Safe Area, he locks the car and goes away. Danilo has some problems with his bank, indeed his credit card was blocked some days ago, therefore his payment for the last PowerEnJoy trip failed and his account has been blocked. The next time Mr. Danilo wants to use his account he has to call the assistance.

4.7 Scenario #7 (IKEA scenario)

Robinson Crusoe has a PowerEnJoy account, he logs in, reserves a car with sufficient battery level nearby his home and drives to IKEA store. He parks the car but he would like to be sure that when he comes back the car is still there waiting for him. So, when the engine turns off, Robinson taps the option PAUSE on OBS and exits from the car. PowerEnJoy system charges Robinson for the trip, reset the bill and locks the car automatically after 40 seconds. After a few hours he goes back to the parking lot and unlocks the car from the app. The system charges Robinson for the pause time and he drives to home. He parks near his home, exits from the car and the system charges him for the trip time and locks the car.

4.8 Scenario #8 (discount for plugging)

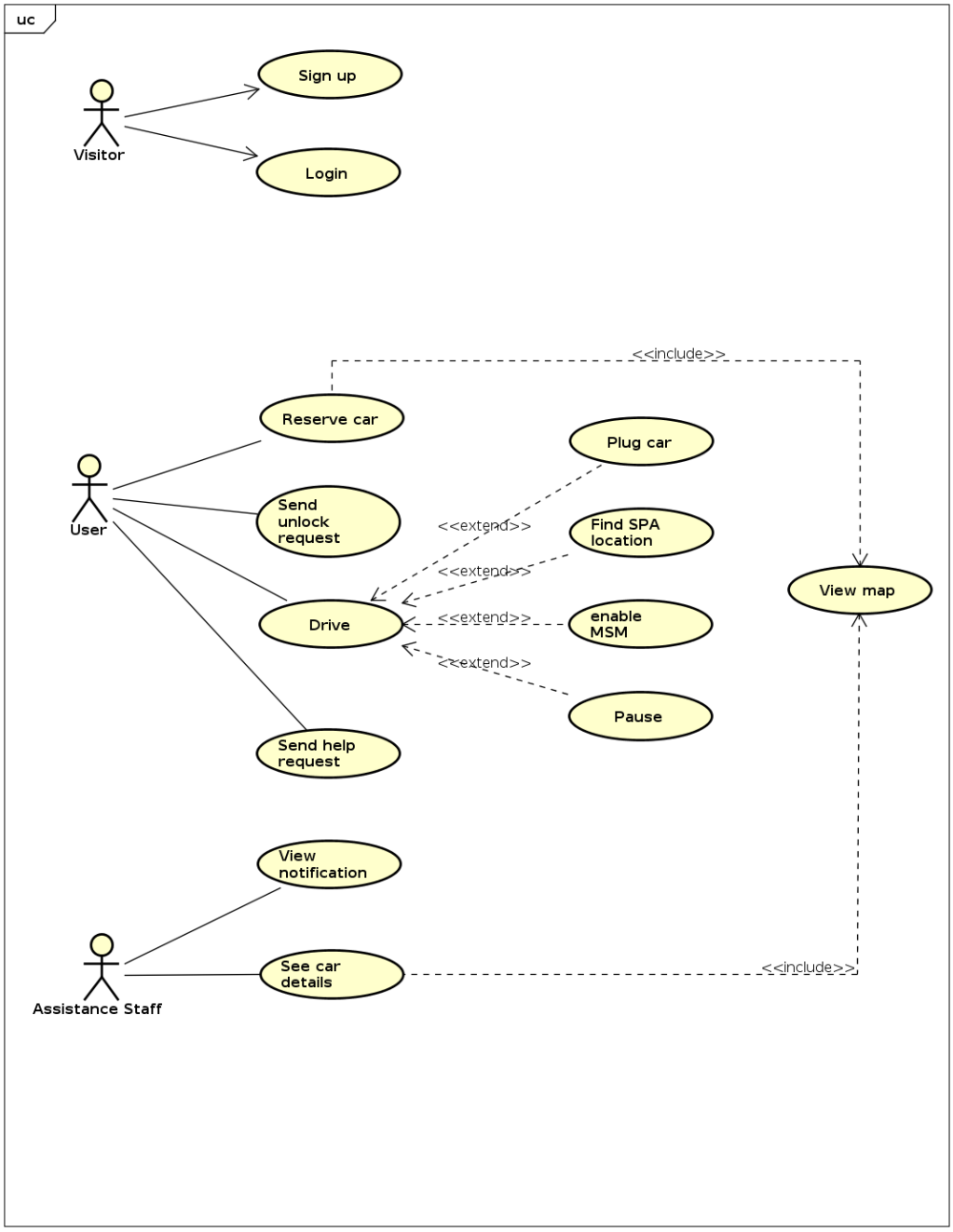
Elon Musk reserves a car via smartphone app and, after unlocking it, drives using the MSM to the nearest SPA. Arrived to the SPA Elon, who is familiar with electric cars, plugs the car to the power grid before the car is locked. Now the car starts charging and TS gives Elon a discount of 30% on his last ride.

4.9 Scenario #9 (car left out of power)

Paola Magnaghi needs to go to Polimi, she logs in and reserves a car from web site. She goes nearby the car and unlocks it by the app, she drives for 30 minutes to reach Polimi and finds a park. She's in late so parks in the first free parking she finds. The battery car is only 15% of the full charge so the System increments her bill 30% and charges her. The System can't set that car as available and notify the assistance. An operator sees on monitor car position and its battery level and goes to recharge the car. When operator finished to recharge the battery, The System set the car as available.

5 UML models

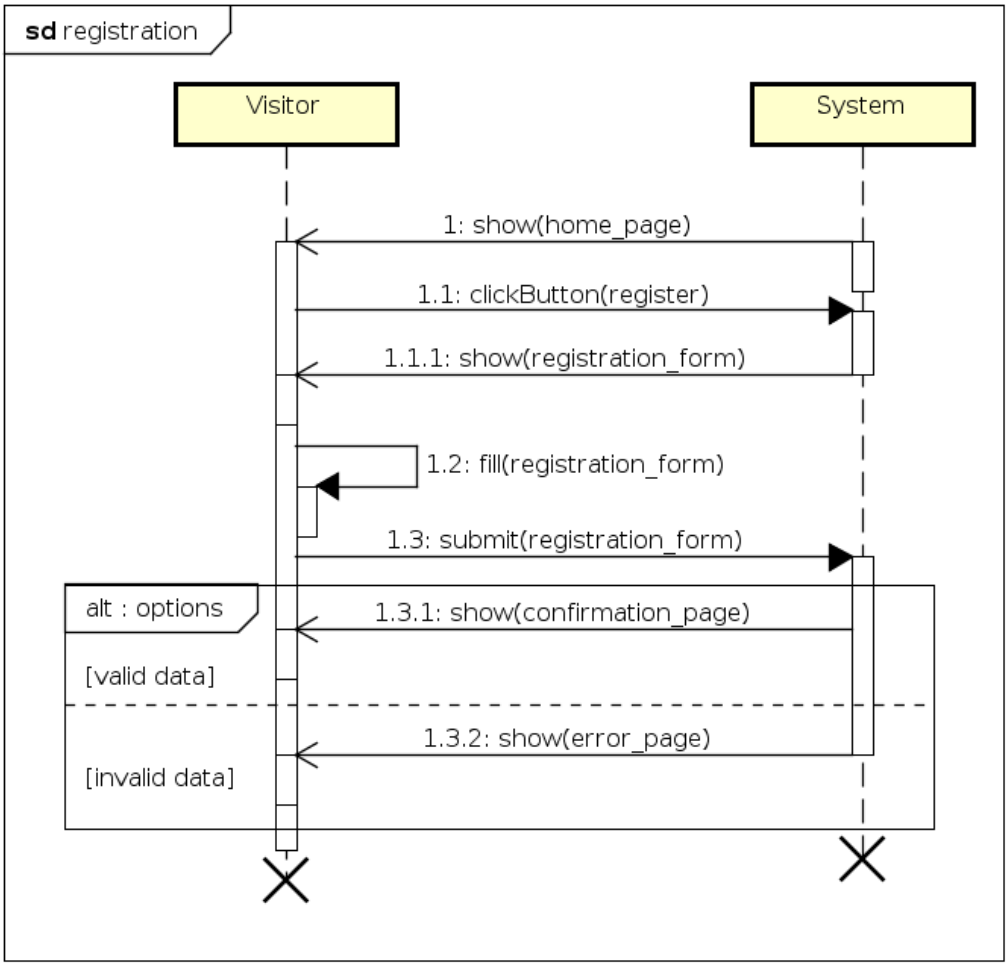
5.1 Use case diagram



5.2 Use case description & sequence diagram

5.2.1 Sign up

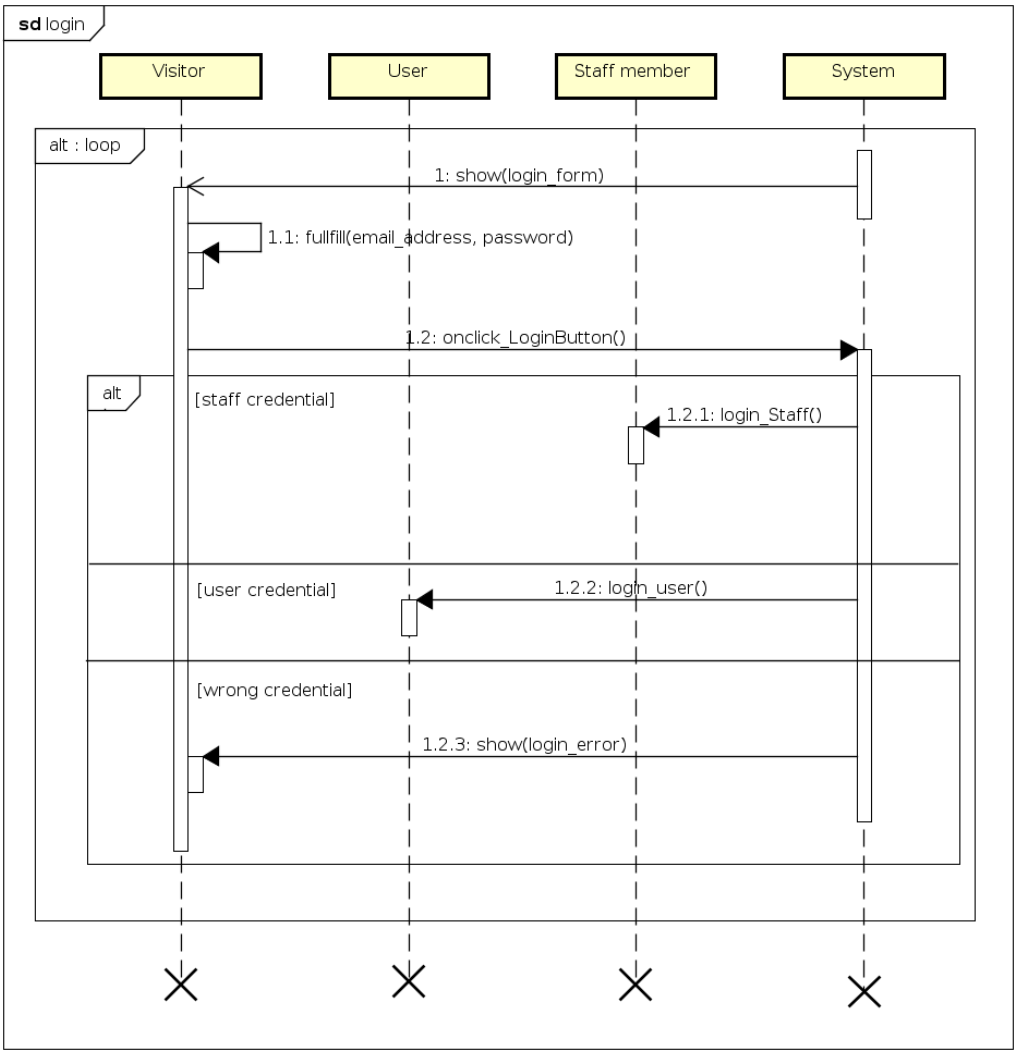
Goal	G1
Actors	Visitor
Assumption	Visitor has a valid driver licence, Visitor is in major age, Visitor has a valid way to pay
Entry conditions	Visitor opens our website or mobile application and clicks on Register button.
Flow of events	1. Visitor fills the form providing all personal and payment data required. 2. Visitor clicks on “Confirm” button. 3. TS checks if the visitor wasn’t already signed up via the number of driver licence. 4. The application saves all data in the DB.
Exit conditions	Visitor data are inserted in the DB, now he can log into the web application
Exceptions	Some data were not inserted, were in an invalid format or the visitor was yet a user of TS



5.2.2 Login

Goal	G2
Actors	Visitor, User, Assistance Staff
Assumption	Visitor is already registered
Entry conditions	Visitor goes to PowerEnJoy login page or PowerEnJoy app
Flow of events	1. Visitor enters his email address and password 2. Visitor clicks the Login button 3. System redirects visitor to customer page if he is an user, to service page if he is a staff member
Exit conditions	Visitor logs in successfully: if he is a customer he access to the user page, if he is a staff member he access to service page

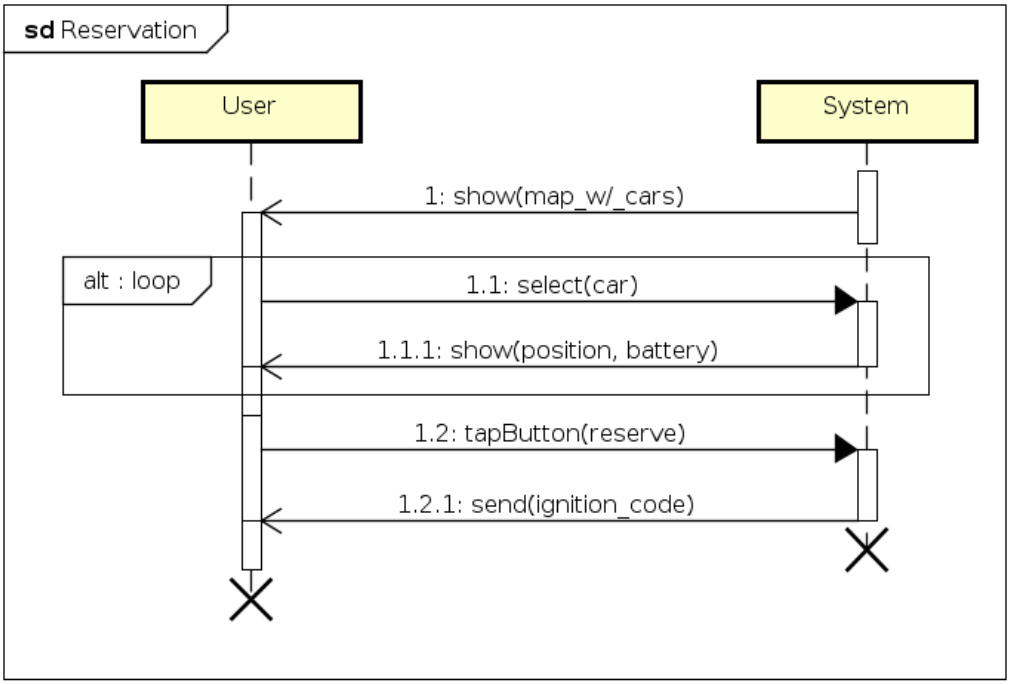
Exceptions	Email address and password combination does not correspond to any existing account



5.2.3 Reserve car

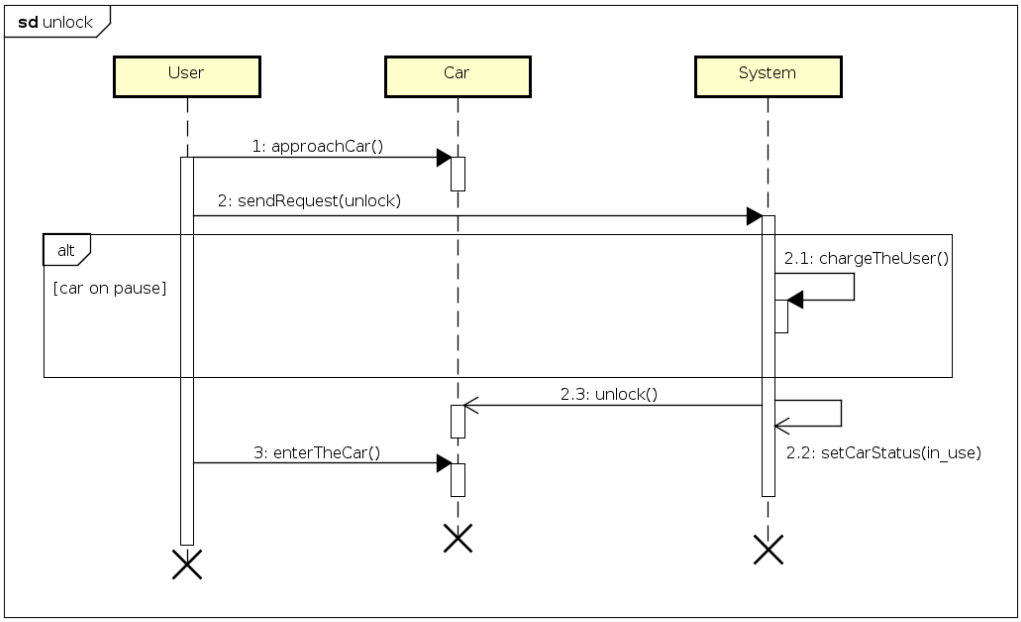
Goal	G3
Actors	User
Assumption	User is in the reservation page or screen
Entry conditions	User views a map with all available cars
Flow of events	1. User selects a car 2. User sees car position and battery level 3. User taps the Reserve button 4. User receives a confirmation message with the ignition code

Exit conditions	User reservation is confirmed
Exceptions	User exits the reservation procedure



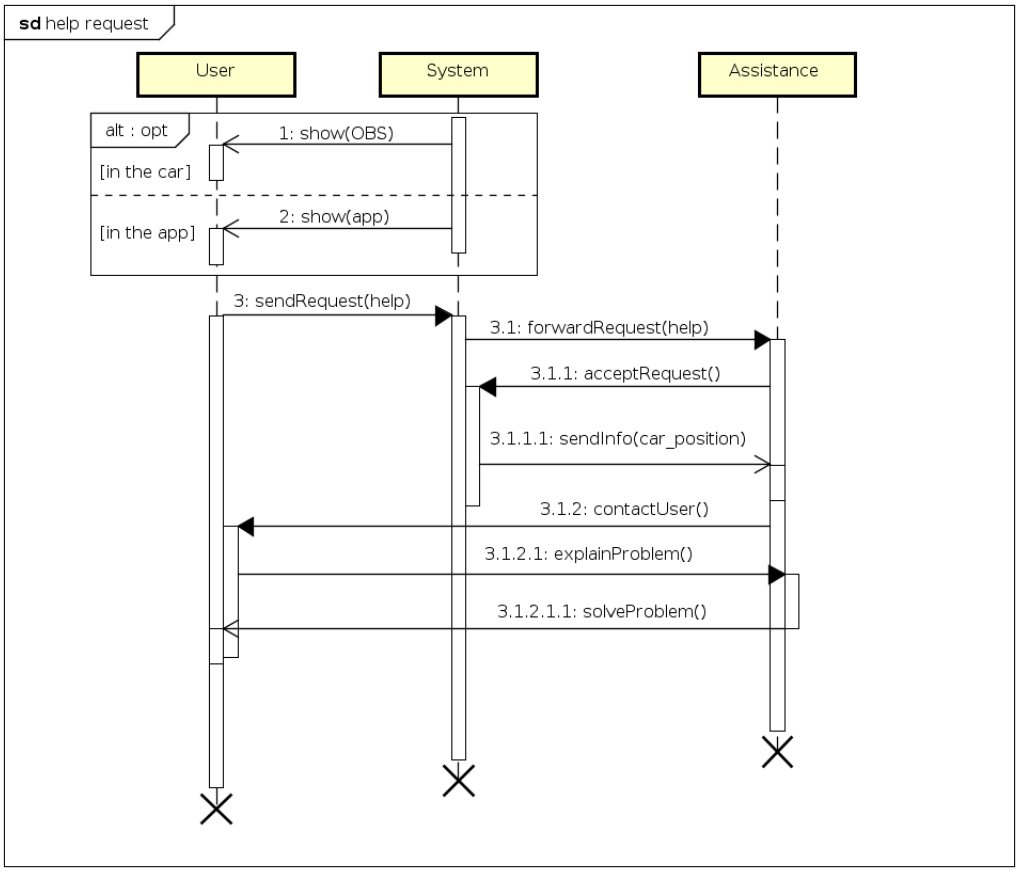
5.2.4 Send unlock request

Goal	G6
Actors	User
Assumption	User has reserved a car or he has a car on PAUSE
Entry conditions	User is on reservation screen on mobile app and he is within 10 meters from the car
Flow of events	1. User clicks the Unlock button
Exit conditions	The car is unlocked
Exceptions	The user is not within 10m of the car and the Unlock button is disabled



5.2.5 Send help request

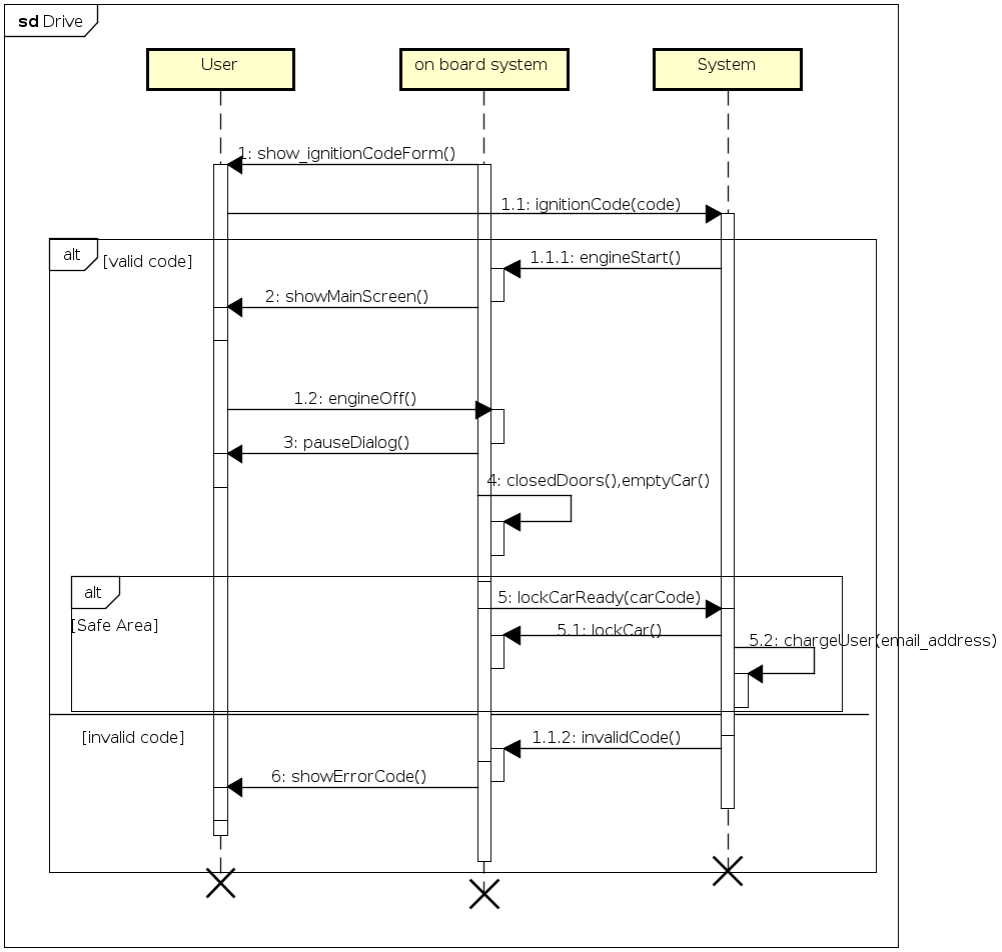
Goal	G13
Actors	User, assistance staff
Assumption	User has the app open or he is in the car
Entry conditions	User taps the Help button from app or OBS
Flow of events	1. Assistance staff answers 2. System shows car position to the assistance if the call came from OBS
Exit conditions	Assistance is now giving support to the user
Exceptions	---



5.2.6 Drive

Goal	G7,G11
Actors	User
Assumption	User has unlocked a car
Entry conditions	User enters ignition code on the OBS
Flow of events	1. Engine is activated 2. User drives to its destination 3. User turns off the engine 4. System on board asks to the user if he wants to do a pause 5. System starts timer when car is empty and doors are closed 6. System locks the car and charges the user
Exit conditions	User has ended his trip
Exceptions	User enters wrong ignition code, User exits from the car while he is outside The

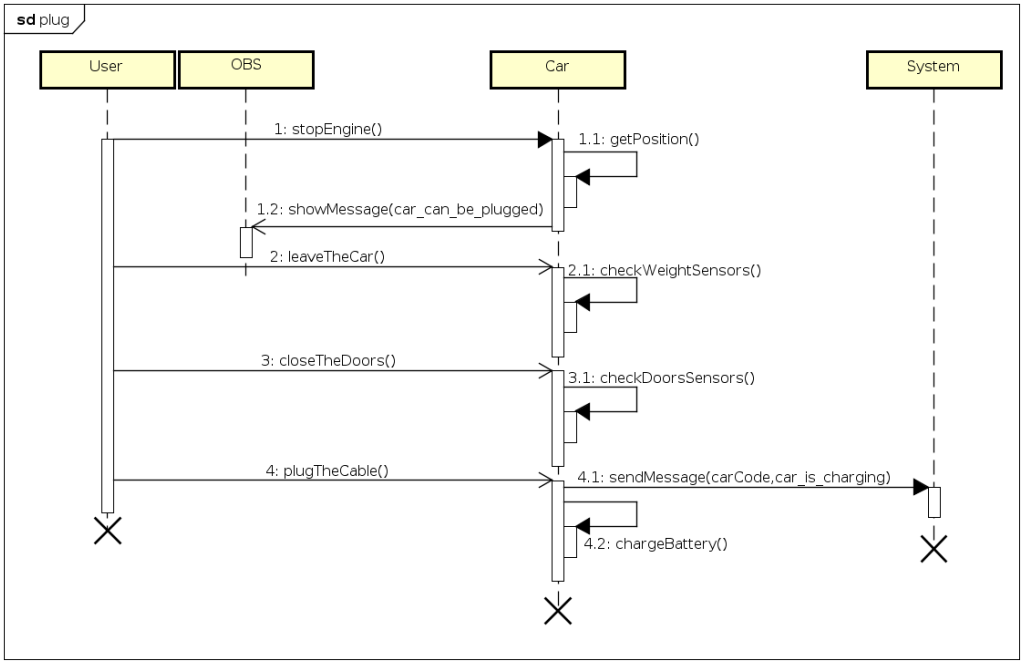
	Area



This use case could be extended by these other use cases:

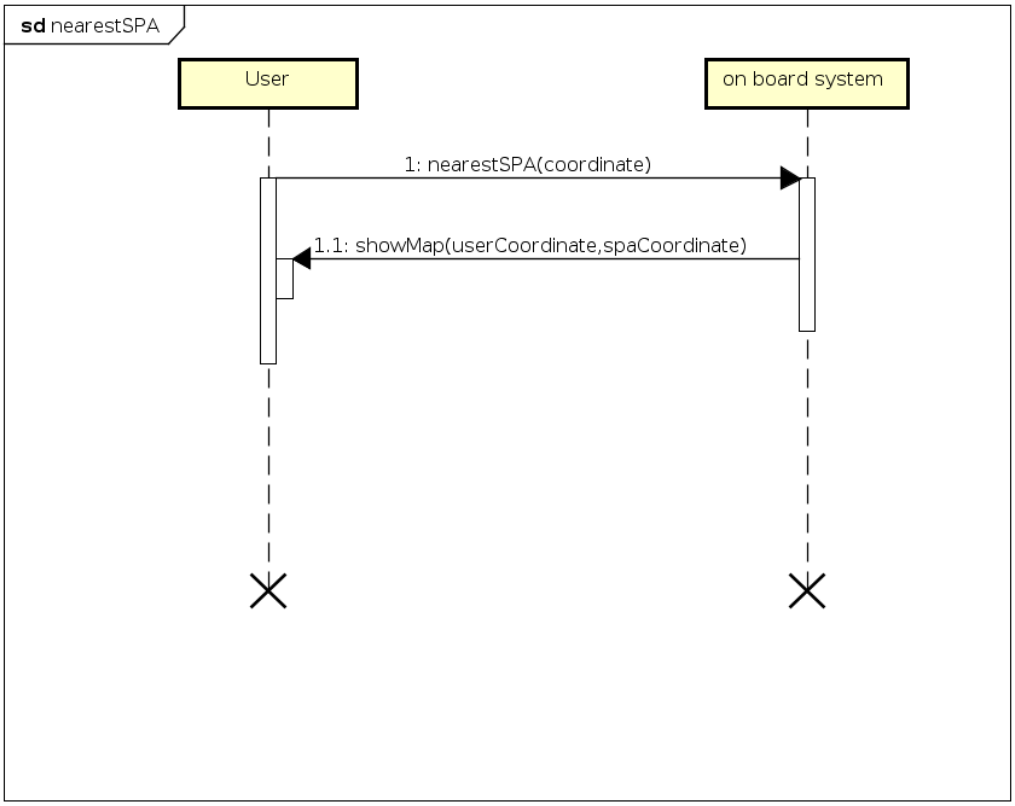
5.2.6.1 Plug car

Goal	G10
Actors	User
Assumption	User has been driving a car
Entry conditions	User has parked a car in a SPA
Flow of events	1. User exits from the car 2. User closes the car doors 3. User takes the charging cable from the charging station and plugs it in the car
Exit conditions	The car is recharging
Exceptions	---



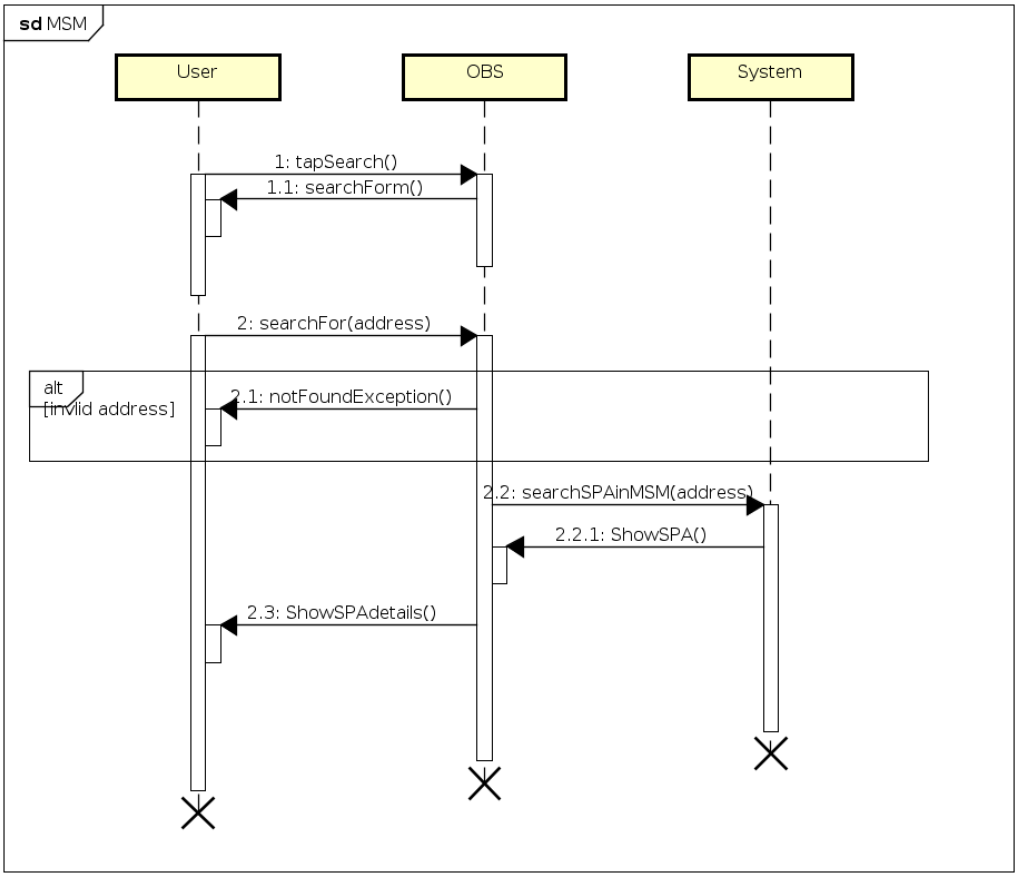
5.2.6.2 Find SPA location

Goal	G8
Actors	User
Assumption	User is in the car
Entry conditions	User is driving
Flow of events	1. User taps on “nearest SPA” on OBS 2. OBS shows the user a map with his position and SPAs position
Exit conditions	The user has seen where SPAs are located
Exceptions	---



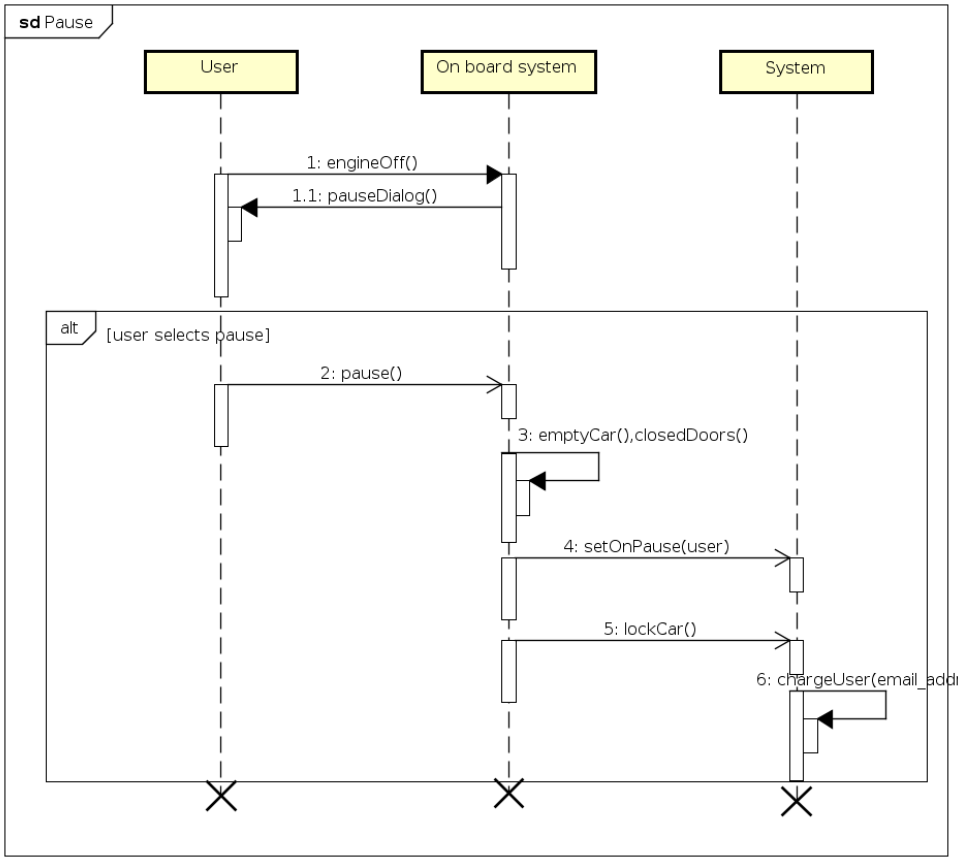
5.2.6.3 Enable MSM

Goal	G9
Actors	User
Assumption	User is in the car
Entry conditions	User taps MSM option
Flow of events	1. User is in the car searches a destination on the OBS map 2. System provides the advised SPA to save money and ensure uniform distribution of cars
Exit conditions	User has the information about SPA that let him save money
Exceptions	Address searched doesn't exist



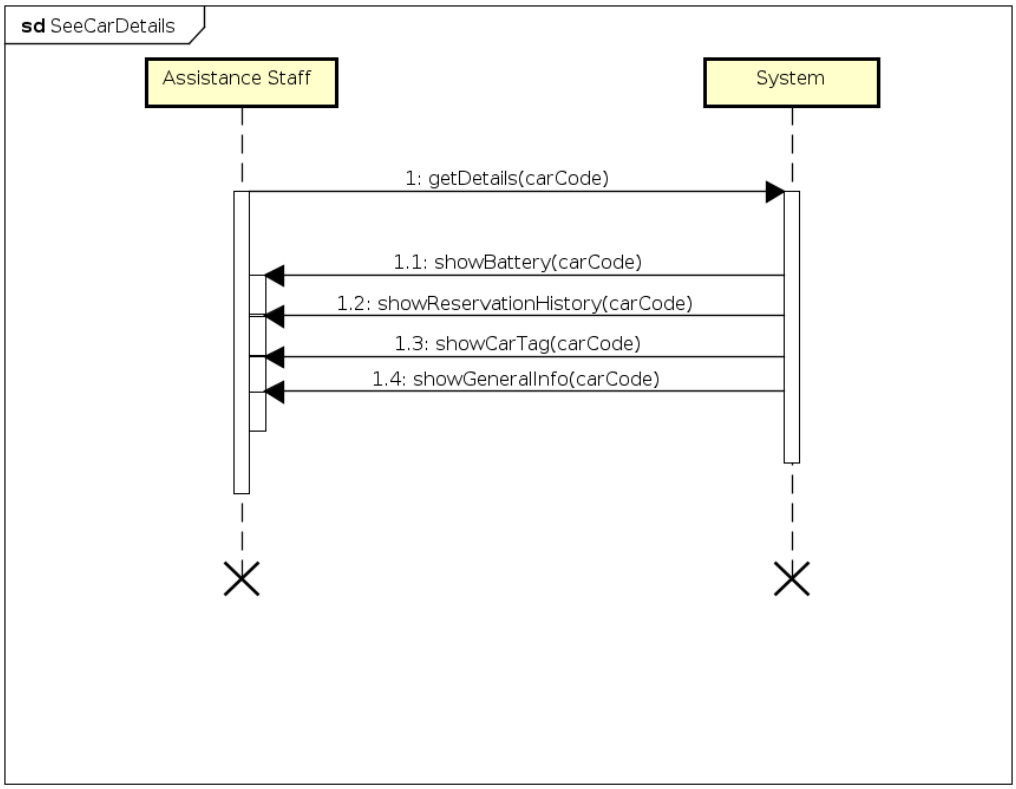
5.2.6.4 Pause

Goal	G12, G11
Actors	User
Assumption	User was driving and turned off the engine in a safe area outside SPA.
Entry conditions	User taps “PAUSE” on the OBS
Flow of events	1. System starts timer when car is empty and doors are closed 2. System locks the car and charges the user
Exit conditions	The car is in PAUSE mode and locked
Exceptions	---



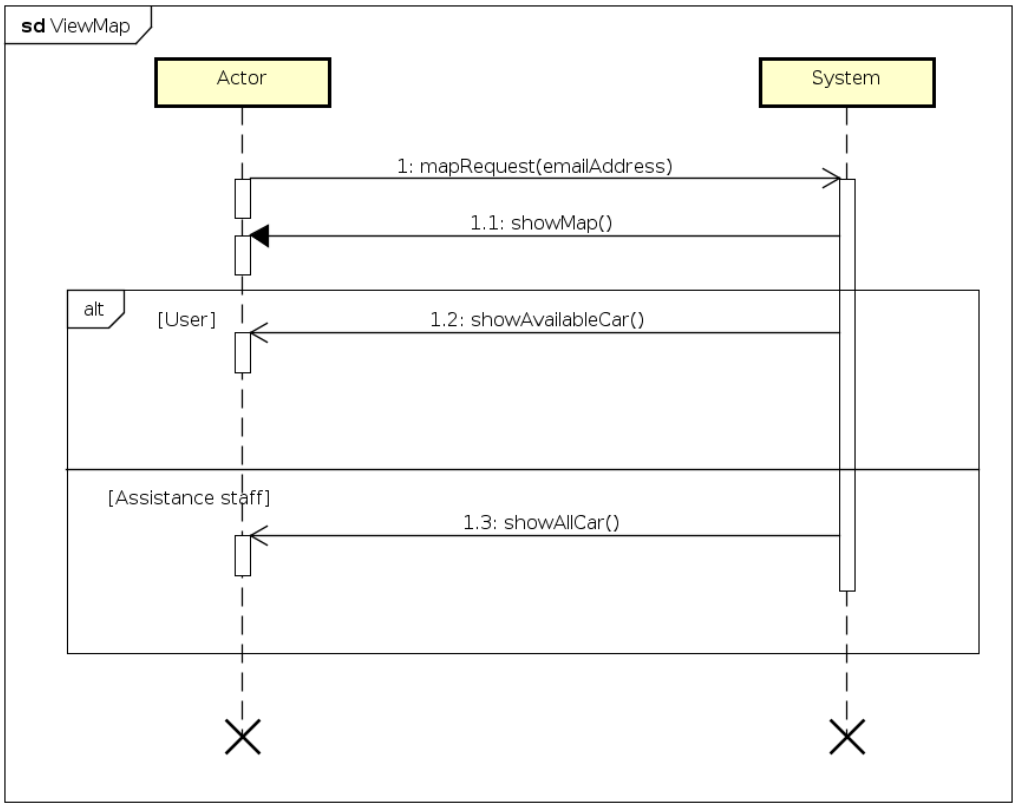
5.2.7 See car details

Goal	G14
Actors	Assistance staff
Assumption	Assistance staff has credentials to access the system
Entry conditions	Staff member viewed the map (4.2.8)
Flow of events	1. Staff member selects a car 2. System shows all car details
Exit conditions	System has shown car details
Exceptions	---

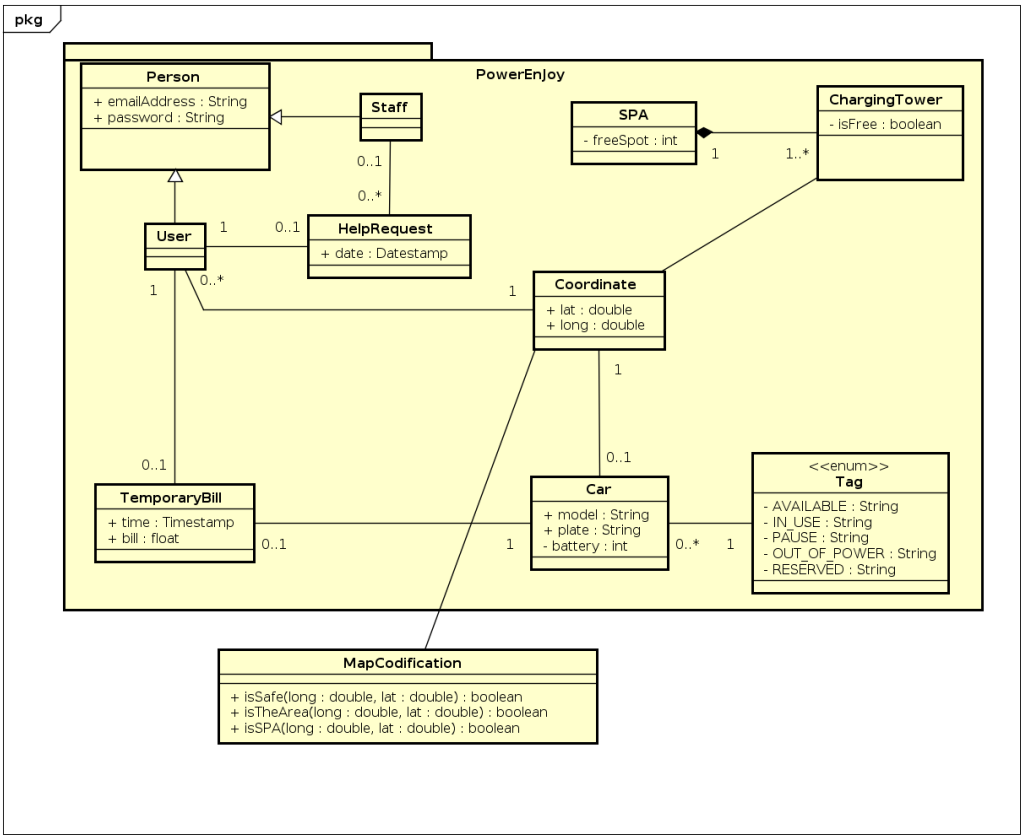


5.2.8 View map

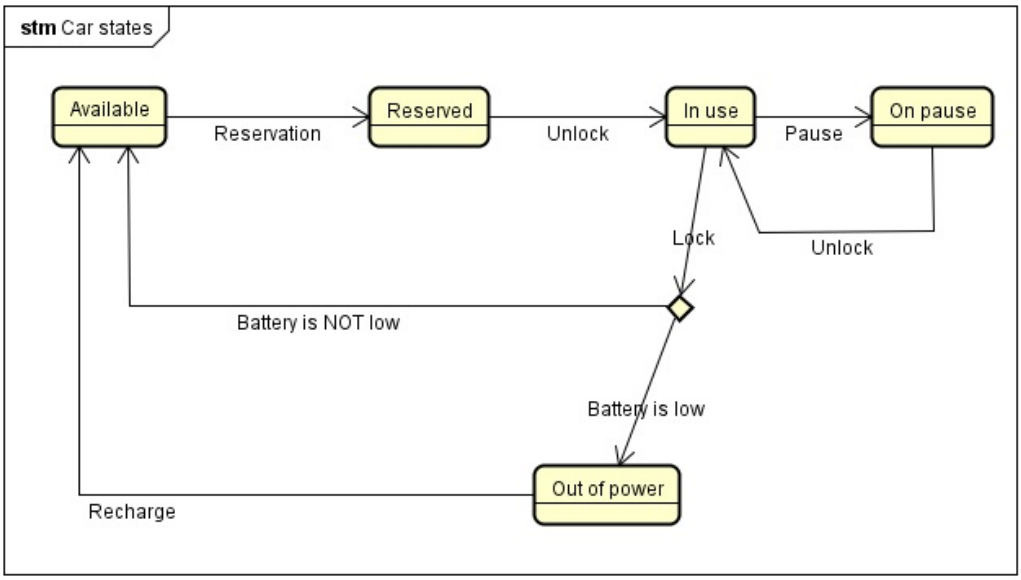
Goal	G3, G14
Actors	User, Assistance staff
Assumption	Actor has credentials to access the system
Entry conditions	Actor accesses the system and click on the Map button
Flow of events	1. Depending on the actor, system shows a map with different tagged cars: 1a. User visualize available cars 1b. Assistance visualize all vehicles
Exit conditions	System has shown the map
Exceptions	---


sd ViewMap
Actor
System
1: mapRequest(emailAddress)
1.1: showMap()
alt
[User]
1.2: showAvailableCar()
[Assistance staff]
1.3: showAllCar()

5.3 Class diagram



5.4 Statechart diagram



6 Alloy modelling

6.1 Alloy code

```
--@@@@@@@@@@@@@ SIGS AND FACTS @@@@@@@@@@@@@@@

abstract sig Person{
   email: one Email
}
sig Email{}
--each person has different email
fact uniquePerson{all p1,p2:Person|(p1!=p2)=>p1.email!=p2.email}
--each email has user
fact EmailPerson{Person.email=Email}

-----------------------------------------------------PLATE
sig Plate{}
--each plate has a car
fact PlateCar{Car.plate=Plate}
--each car has different plate
fact uniquePlate{all c1,c2:Car | (c1!=c2) => c1.plate!=c2.plate}

-----------------------------------------------------COORDINATE
sig Coordinate{
   area: one Area,
   spa: one ServicePA,
}
fact {Coordinate=User.coordinate+ChargingTower.is+Car.is}

-----------------------------------------------------AREA
enum Area{SafeArea,noSafeArea,OutsideArea}

-----------------------------------------------------STAFF
sig Staff extends Person{
   handle: set HelpRequest
}
fact UniqueHandle{all s1, s2:Staff | no h:HelpRequest | s1!=s2 and (h in s1.handle) and (h in s2.handle)}
```

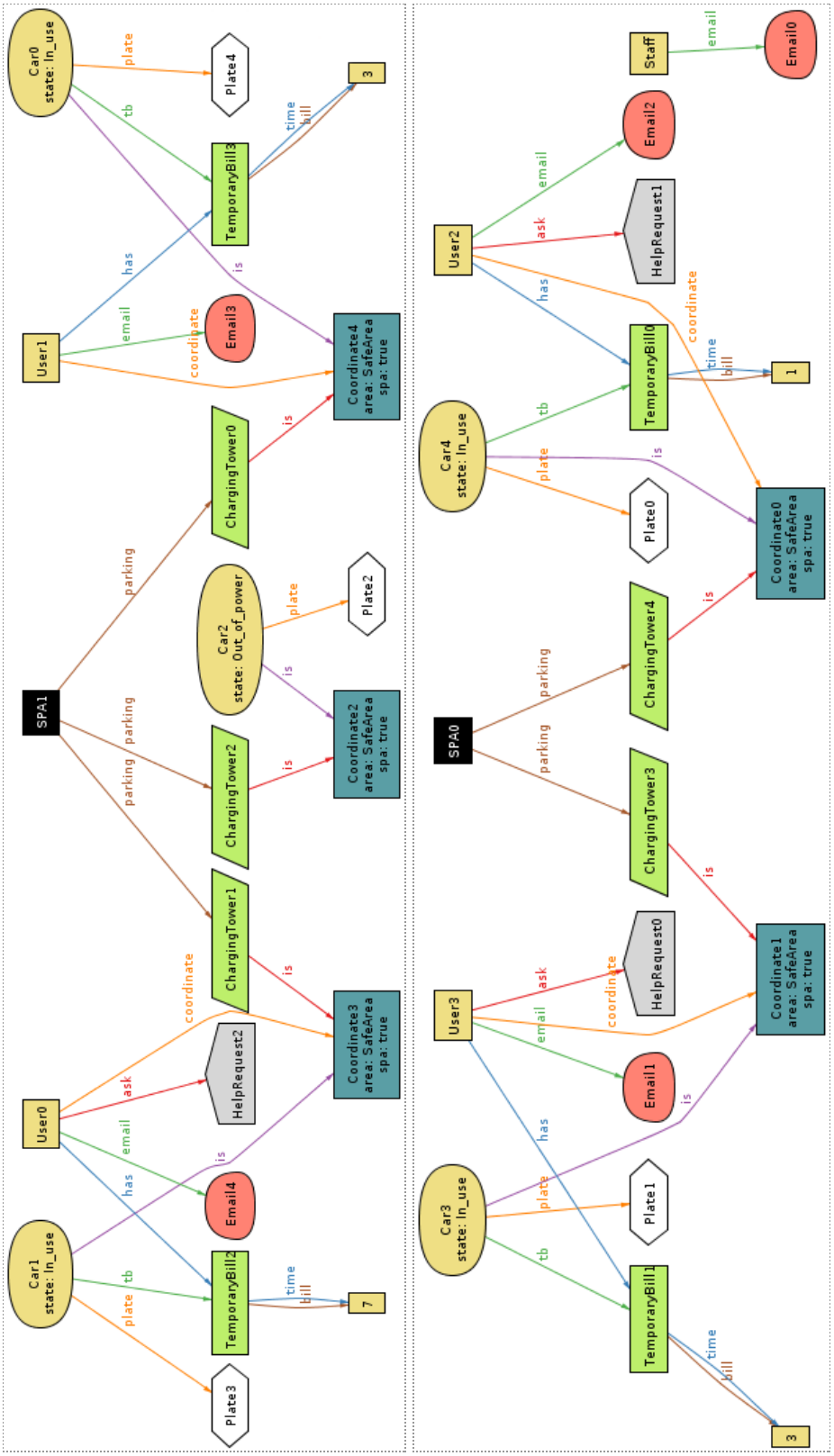
```
-------------------------------------------------------USER
sig User extends Person{
   has:lone TemporaryBill,
   ask: lone HelpRequest,
   coordinate: one Coordinate
}
--different users have different reservation
fact UniqueReservation{no u1,u2:User | u1!=u2 and u1.has=u2.has}
--each user has different coordinate
fact UserUniquePosition{ no u1,u2:User | u1!=u2 and u1.coordinate=u2.coordinate}

-------------------------------------------------------CAR
sig Car{
   plate: one Plate,
   state: one Tag,
   is: one Coordinate,
   tb: lone TemporaryBill
}

--each car has different coordinate
fact CarCoord{no c1,c2:Car | c1!=c2 and c1.is=c2.is}
--all available and reserved cars  are in safe
fact AvCarSafe{all c:Car | (c.state=Available or c.state=Reserved or c.state=Out_of_power) => c.is.area=SafeArea}
--all pause cars are in safe or outside the area
fact PauseCarSafeOut{no c:Car | (c.state=Pause) and (c.is.area=SafeArea and c.is.spa=true ) }
--if car position is in user position, his car is in use
fact UserUseCar{all u:User | all c:Car | u.coordinate=c.is iff (c.state=In_use and c.tb=u.has)}
--each use or pause car has reservation

----QUI PROBLEMI
fact TBcar{all c:Car| no  t:TemporaryBill | c.state=Available and c.tb=t}
fact TBcar{all c:Car| no  t:TemporaryBill | c.state=Out_of_power and c.tb=t}
--each available or oop car has reservation
fact noTBcar{ no c:Car| (c.state=In_use or c.state=Pause or c.state=Reserved) and #c.tb=0}

-------------------------------------------------------TEMPORARY BILL
enum Tag{Available,Pause,Out_of_power,In_use,Reserved}
sig TemporaryBill{
  time: one Int, --minute
  bill:one Int
}{
  time>0
}
fact UniqueReservation2{no c1,c2:Car | c1!=c2 and c1.tb=c2.tb}
--tempBill
fact TempBillUser{User.has=TemporaryBill}
fact TempBillCar{TemporaryBill=Car.tb}
fact BillFee{ all c:Car | c.state=Reserved => (c.tb.bill=0)}
--fact TimeReserved{ all c:Car | c.state=Reserved => (c.tb.time<60)}
fact BillFee2{ all c:Car | (c.state=In_use or c.state=Pause) => (c.tb.bill=mul[c.tb.time,1])} -- 1eur  per minute fee

-------------------------------------------------------HELP REQUEST
sig HelpRequest{}
--help request
fact HelpRequestUser{User.ask=HelpRequest}
fact HelpRequestStaff{Staff.handle in HelpRequest}
fact UniqueHelpRequest{no u1,u2:User | u1!=u2 and u1.ask=u2.ask}


-------------------------------------------------------SPA
enum ServicePA{true,false}
sig SPA{parking: some ChargingTower}
--spa is always in safe area
fact SPASafe{all c:Coordinate| c.spa=true => c.area=SafeArea}

-------------------------------------------------------CHARGING TOWER
sig ChargingTower{
   is:one Coordinate
}
enum SpotFree{True,False}
fact UniqueCoordinateCT{no c1,c2:ChargingTower | c1!=c2 and c1.is=c2.is}
--every chargingTower has in a SPA
fact chargingTowerSPA{all c:ChargingTower | one s:SPA | c in s.parking}
--all charging towers are in SPA
fact ChargingTower{all c:ChargingTower | c.is.spa=true}
fact{all coo:Coordinate|one crg:ChargingTower|coo.spa=true implies crg.is=coo}

--@@@@@@@@@@@@@@@   ASSERTS   @@@@@@@@@@@@@@@@@@

--all oop or available cars are not linked to tbills
assert BillForPauseOrUseOrReserved{ all car:Car |no t:TemporaryBill | car.tb=t and (car.state=Out_of_power or car.state=Available) }
--available cars are in safe area
assert AvailableOnlyInSafeArea{all c:Car | c.state=Available implies c.is.area=SafeArea}
--doesnt exist available car outside the area
assert NoAvailableCarOutOfTheArea{no c:Car | c.state=Available and c.is.area=OutsideArea}
--Bill remains 0 if car is Reserved
assert BillReserved{all c:Car|no t:TemporaryBill | c.tb=t and c.state=Reserved and t.bill!=0}
--to this abstract level world bill cannot be 0 if car is in use or in pause
assert BillOtherwise{all c:Car|no t:TemporaryBill | (c.state=In_use or c.state=Pause)and c.tb=t and t.bill=0}
--available cars are inside The Area
assert AvailableIsInside{no c:Car | c.state=Available and c.is.area=OutsideArea}
```

```
--@@@@@@@@@@@@@@@   PREDICATES  @@@@@@@@@@@@@@@

pred show{
#Car>3
#User>0
#Staff>1
#SPA>1
#ChargingTower>#SPA
}

pred show2{
(some c:Car|c.state=Available)
(some c2:Car|c2.state=Pause)
(some c3:Car| c3.state=Reserved)
(some c4:Car| c4.state=In_use)
(some c5:Car | c5.state=Pause)
}

pred HelpRequest(u:User, s1,s2:Staff){
   u.ask not in s1.handle implies s2.handle=s1.handle+u.ask
}

--@@@@@@@@@@ COMMANDS @@@@@@@@@@@@@@@@@

check NoAvailableCarOutOfTheArea    -- #1
check BillForPauseOrUseOrReserved   -- #2
check AvailableIsInside             -- #3
check AvailableOnlyInSafeArea       -- #4
check BillReserved                  -- #5
check BillOtherwise                 -- #6
run HelpRequest                     -- #7
run show for 5                      -- #8
run show2 for 10                    -- #9
```

```
9 commands were executed. The results are:
   #1: No counterexample found. NoAvailableCarOutOfTheArea may be valid.
   #2: No counterexample found. BillForPauseOrUseOrReserved may be valid.
   #3: No counterexample found. AvailableIsInside may be valid.
   #4: No counterexample found. AvailableOnlyInSafeArea may be valid.
   #5: No counterexample found. BillReserved may be valid.
   #6: No counterexample found. BillOtherwise may be valid.
   #7: Instance found. HelpRequest is consistent.
   #8: Instance found. show is consistent.
   #9: Instance found. show2 is consistent.
```

6.2 Alloy worlds

In this paragraph we will show the world generated by the predicate “show”. The two images represent the same instance split into two parts.



7 Used tools

The tools we used to create this RASD document are:

- Alloy analyzer 4.2
- Pencil for mockups
- Google Docs for collaborative real-time editing of this document
- Astah Professional for UML diagrams

8 Hours of work

8.1 Davide Moneta

21/10/2016 3h00'
26/10/2016 1h00'
27/10/2016 5h00'
31/10/2016 5h00'
2/11/2016 1h00'
7/11/2016 2h00'
9/11/2016 3h30'
10/11/2016 3h30'
11/11/2016 2h00'
12/11/2016 2h30'
13/11/2016 4h30'

8.2 Federico Oldani

20/10/2016: 1h30'
21/10/2016: 2h30'
23/10/2016: 1h00'
24/10/2016: 0h45'
25/10/2016: 1h30'
27/10/2016: 2h00'
28/10/2016: 2h30'
30/10/2016: 4h00'
31/10/2016: 1h00'
2/11/2016: 2h30'

3/11/2016: 2h00’
4/11/2016: 4h00’
5/11/2016: 5h00’
6/11/2016: 2h00’
7/11/2016: 2h30’
8/11/2016: 3h00’
9/11/2016: 3h00’
10/11/2016: 5h00’
11/11/2016: 1h30’
12/11/2016: 7h00’
13/11/2016: 3h00’

8.3 Luca Oppedisano

20/10/2016: 1h30’
21/10/2016: 1h30’
24/10/2016: 0h45’
27/10/2016 5h00’
28/10/2016 1h30’
29/10/2016 3h00’
30/10/2016 2h30’
31/10/2016 4h30’
01/11/2016 2h00’
2/11/2016 3h:45’
3/11/2016 2h00’
4/11/2016 5h00’
6/11/2016 5h00’
7/11/2016 3h30’
8/11/2016 2h00’
9/11/2016 3h00’
10/11/2016 6h30’
11/11/2016 2h00’
13/11/2016 3h00’

9 Changelog

- Replace class diagram with the correct one